

동적 프로그래밍 기법을 이용한 효율적인 배치 개선 알고리즘

오은경*, 허성우**

동아대학교 컴퓨터공학과

e-mail : * ekoh@mail.donga.ac.kr, ** swhur@daunet.donga.ac.kr

An Efficient Algorithm for Improving the Detailed Placement Using Dynamic Programming Technique

Eun-Kyung Oh*, Sung-Woo Hur**

Dept. of Computer Engineering, Dong-A University

요 약

VLSI 칩 설계에서 물리 설계의 과정은 칩의 크기나 성능 그리고 칩 생산수율 등에 결정적인 영향을 미치는 매우 중요한 단계로서 이 자체가 매우 복잡하기 때문에 또 여러 세부 단계로 나누어 물리 설계가 수행된다. 그러므로 물리 설계를 위한 많은 연구가 계속되어 왔고 그 중 배치를 위한 연구도 20 여년 이상 되어 왔다. 좋은 배치를 얻기 위해 크게 두 단계, 즉 광역배치(global placement)와 미세배치(detailed placement) 단계로 나누어 수행되는데 본 논문에서는 미세 배치를 매우 효과적으로 개선할 수 있는 동적 프로그래밍 기법을 이용한 효율적인 알고리즘을 제시하며 실험 결과를 통하여 제시된 알고리즘의 효율성을 입증하였다. 최근의 최신 배치프로그램을 통해 얻은 결과에 본 논문에서 제시한 알고리즘을 적용했을 경우 HP(half perimeter)가 평균 3.4 % 정도 개선되었다

1. 서론

배치문제는 VLSI 칩 설계 시 회로의 성능, 칩의 면적, 칩 생산수율(yield), 배선 능력 등에 결정적인 영향을 미친다. 예를 들어, 배선 길이를 줄이려는 목표로 배치문제를 고려한다고 하자. 그러면, 배선의 길이가 짧을수록 배선에 필요한 공간이 줄어들게 되어 전체 레이아웃 면적이 작아지는 효과가 있고, 신호 지연이 감소되므로 성능에도 다소 긍정적인 영향을 미치게 된다. 또한 레이아웃 면적이 줄어들면, 한 웨이퍼에 많은 칩을 제조할 수 있어 칩 생산수율이 높아진다. 또한 배선 길이를 줄이는 것이 배선 가능성을 높이는 것보다도 매우 밀접한 것으로 최근의 연구는 보여 준다. 즉, Wang 과 Sarrafzadeh 는 배선길이를 줄이는 것이 전역적으로 볼 때 배선 밀집도(congestion)를 낮추는 효과가 있음을 실험적으로 관찰했다[1]. 그러기 때문에 대부분의 배치문제는 명시적으로 배선길이를 줄이려는 목표를 추구하고 있다.

배치문제를 위한 접근 방법은 크게 세가지로 분류될 수 있는데 첫째는 분할 기법을 이용한 방법으로써

회로와 칩 영역을 하향식으로 계속 분할해 감으로써 각 셀의 위치를 결정하고자 하는 방법이다[2-5], 두번째 접근방법은 force-directed 방법으로 알려진 해석적 기법을 사용하는 것이다. 각 셀의 연결도에 대한 정보를 이용하여 셀의 안정된 위치를 찾고, 중복을 해결함으로써 최종 배치를 얻는 방법이다[6-9]. 세번째 방법으로는 simulated annealing 기법이라고 알려진 것으로서 일단 주어진 배치를 바탕으로 특정한 두 셀의 위치를 맞 바꾸거나, 한 셀의 위치를 옮기는 작업 등을 반복함으로써 배치의 성능을 개선해 가는 방법이다[10,11]. 최근에 와서는 앞의 기법들을 복합적으로 사용한 하이브리드 알고리즘들이 발표되고 있으며, 결과는 단순히 한 방법을 사용한 것보다 우수한 것으로 보여진다 [12-14].

최근에 발표된 대부분의 배치 휴리스틱은 첫 단계에서 광역배치를 얻고, 그 다음 단계에서 미세배치를 개선해 가는 기법을 주로 사용하고 있다. 좋은 광역배치를 구하는 것이 궁극적으로 좋은 배치를 얻는 필수과정이기 때문에 광역배치를 위해 여러가지 기법을

사용한다. 또한 광역배치가 좋다 하더라도 광역배치는 많은 중복을 허용하고 있기 때문에 중복을 어떻게 해결할 것인가 하는 방법에 따라 미세배치의 질이 크게 영향을 받는다.

배치의 목적이 구체적으로 무엇이든 목시적으로는 배선길이를 최소화 하는 목적이 포함되어 있기 때문에 대부분의 연구가 추정 배선 길이를 최소화 하는데 초점을 맞추고 있다. 본 논문에서는 최종적인 미세배치를 더 개선할 수 있는, 즉 추정배선길이를 최소화 할 수 있는 매우 효과적이고 빠른 알고리즘을 제시하고자 한다.

배선 길이를 추정하기 위해 효율적이면서 널리 사용되고 있는 방법은 그림 1 에서 보인 것 처럼 HP (half-perimeter) 모델을 사용하는 것이다. HP 는 그 넷에 관계된 모든 핀을 둘러싸는 직사각형을 구한 후, 둘레 길이의 반을 계산한 것이다. 이것은 핀 수가 2 또는 3 인 넷에서 돌아가는 배선(detour)이 없을 경우 정확한 길이이고, 핀수가 4 이 이상인 경우 배선길이의 lower bound 이다. 실제 회로에선 대부분의 넷이 2 혹은 3 개의 핀으로만 구성되어 있기 때문에 배선길이를 추정하기 위해 HP 를 사용해도 큰 오차가 없다. 각 핀은 셀의 중앙에 위치한다고 가정한다.

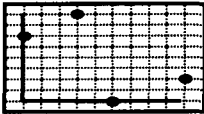


그림 1. 핀 수가 4 인 넷의 HP=17

2. 배치 개선 알고리즘

2.1. 알고리즘의 개요

제안하는 알고리즘의 개괄적인 동작 원리는 다음과 같다. 주어진 미세배치를 이용하여 각 행에 위치한 셀의 위치를 그 행 내에서 재조정함으로써 배선길이를 줄이는 것이 목표이다. 전체적인 알고리즘은 그림 2 와 같다.

입력: 배치, 윈도우 크기 W
 출력: 개선된 배치

1. 주어진 윈도우 크기 W 에 대해, 하나의 행 내에 있는 셀들로부터 W 범위 안에서 부분열 A 를 찾아 낸다
2. A 셀들에 포함되지 않은 셀들은 B 에 포함한다. 이 때 A 와 B 배열 순서는 현 배치의 상대적 순서를 유지한다
3. 두 부분 배열 A 와 B 내에 있는 셀들의 상대적 위치를 유지하면서 A 와 B 를 인터리브(interleave)시킨다.
4. 단계 1~3 을 첫째 행부터 마지막 행까지, 그리고 각 행내에서는 윈도우를 우측으로 이동하면서 반복 적용한다
5. 위의 모든 단계를 배치가 개선되지 않을 때 까지 반복한다

그림 2. 배치개선 알고리즘

2.2. 최적 인터리빙 (Optimal Interleaving)

알고리즘 “배치개선”의 핵심은 단계 3 의 과정이다. 단계 3 은 최적 인터리빙이라 불리는 기법을 통해 가능하며 이의 기본적인 동작은 그림 3 과 같이 표현될 수 있다. 그림 3 은 부분 배열 A, B 내에서의 배열 순서는 그대로 유지한 상태에서 재조합 되는 것을 보여준다. 그림 3 에서는 윈도우의 크기가 7 인 경우이며, 부분 배열 A 와 B 에는 각각 3 개의 셀과 4 개의 셀로 분할되었다. 기본적으로 A, B 의 분할은 무작위로 행해진다. 일단 각 부분 배열이 얻어지면 이를 인터리브 함으로써 새로운 개선된 셀 순서를 얻게된다..

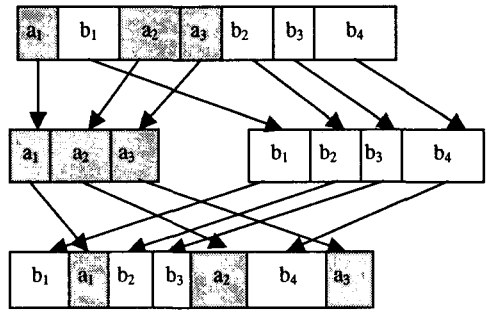


그림 3. 최적 인터리빙의 동작 원리

부분 배열 A, B 의 크기가 각각 n, m 이라면 최적 인터리빙을 통해 고려할 수 있는 모든 가능한 순서의 수는 $\binom{n+m}{n}$ 이 된다. 즉, 최적 인터리빙을 통해 검색 되는 해 공간이 n 과 m 의 크기가 비슷하다면 지수적이라고 말할 수 있다.

두 부분열의 최적 인터리빙은 다이나믹 프로그래밍을 통해서 효율적으로 실행 될 수 있다.

$A = a_1, a_2, a_3, \dots, a_n, B = b_1, b_2, b_3, \dots, b_m$ 라고 하자. S_{ij} 를 $a_1, a_2, a_3, \dots, a_i (i \leq n)$ 와 $b_1, b_2, b_3, \dots, b_j (j \leq m)$ 로부터 얻은 최적 순서라고 하고, $C(S_{ij})$ 는 S_{ij} 의 비용이라고 하자. 여기서 비용이란 부분열로부터 얻은 배선길이의 총합을 의미한다.

A 와 B 의 인터리빙의 목적은 $S_{n,m}$ 을 찾는 것이다. S_{ij} 의 부분적 배치의 비용 $C(S_{ij})$ 는 윈도우 사이즈 W 안에서 S_{ij} 에 속한 셀들의 x 값의 합이다. 이것은 S_{ij} 로 커버된 부분 윈도우의 wiring 조밀도의 일반화된 합으로 보여질 수 있다. 앞선 S_{ij} 의 최적 인터리빙은 윈도우에서 부분열의 셀들의 순서에 독립적이라는 것이 이 아이디어를 적용할 수 있는 키 포인트가 된다. 이 독립성은 동적 프로그램으로의 접근을 가능하게 한다.

설명한 동적프로그래밍을 위한 재귀 관계(recurrence relation)는 다음과 같이 나타내어진다.

$$S_{0,0} = 0$$

$$C(S_{0,0}) = 0$$

$$S_{i,j} = \begin{cases} S_{i-1,j}, & \text{if } C(S_{i-1,j}, a_i) < C(S_{i,j}, b_j) \\ S_{i,j}, & \text{otherwise} \end{cases}$$

2.3. 자료 구조

논문에서 제시하는 최적 알고리즘을 수행하기 위한 자료 구조를 살펴보면 다음과 같다.

최적 인터리빙을 수행하기 위해서는 $S_{i,j}$ 에 대한 정보를 기억할 수 있어야 하며, 그래야 $S_{n,m}$ 을 구할 수 있다. $S_{n,m}$ 은 $(n+1) \times (m+1)$ 인 2차원 배열(표)을 이용하여 얻어진다. 윈도우 안에서 만들어 질 수 있는 조합 중 국부적인 영역안에서의 최소 배선 길이를 가지는 조합만 다음 표의 조합에 사용되므로 각 단계에서의 배선 길이는 독립적이며 모든 경우의 배치를 고려하지 않더라도 효율적인 배치를 할 수 있다. 그림 4에서는 $|A|=3, |B|=4$ 인 경우에 대해 어떻게 $S_{3,4}$ 를 구할 수 있는지 예를 보여준다. C 로 표시된 항의 경우, 원하는 순서는 $S_{2,2}$ 인데 이는 마지막 노드가 a_2 가 되는 경우와 b_2 가 되는 경우 두가지인데 이는 재귀적 관계에서 보듯이 앞에서 구한 $S_{1,2}$ 와 $S_{2,1}$ 을 이용하여 $S_{1,2}a_2$ 와 $S_{2,1}b_2$ 의 배선 길이를 비교함으로써 구할 수 있다. 이와 같은 방법으로 표의 각 정보를 얻어가면 궁극적으로 $S_{3,4}$ 를 구할 수 있고 이는 최적 인터리빙에 의한 새로운 순서에 대한 정보를 제공한다.

		a_1	a_2	a_3
	b_1	$b_1 a_1$	$b_1 a_1 a_2$	$b_1 a_1 a_2 a_3$
	b_2	$b_1 b_2$	$b_1 a_1 b_2$	C
	b_3	$b_1 b_2 b_3$		
	b_4			F

그림 4. 최적 인터리빙을 위한 표

$S_{i,j}$ 로부터 $S_{i+1,j}$ 또는 $S_{i,j+1}$ 을 구하기 위해서는 a_i 또는 b_j 에 관련된 넷 정보를 알고 있어야 되는 것은 물론, 각 넷의 경계에 대한 정보를 가지고 있어야만 정확한 배선길이를 계산 가능하다. 윈도우에 포함된 모든 셀들에 관계된 각 넷에 대해 그 넷의 경계 정보는 그림 5 에서처럼 관리된다. 즉, 각 분할에 대해 좌우 경계위치를, 그리고 다른 행에 있는 셀에 관계되어 있으면서 그 경계치가 고려중인 윈도우의 경계 내에 포함되어 있을 경우 그 좌우 경계치에 대한 값을 알고 있어야 각 셀의 위치를 정할 때 정확한 배선 길이 계산이 가능하다. 그림 5 는 각 넷이 가지고 있는 경계 정보에 대해 개괄적용을 보인 것으로 그림에서 X 로 표시된 것은 다른 행에 있는 셀의 위치를 의미한다.

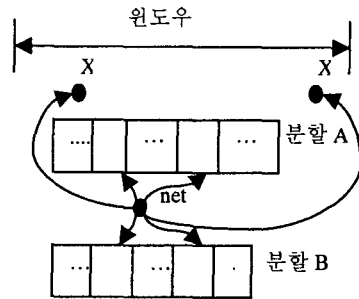


그림 5. 각 넷이 갖고 있는 경계 값 정보

3. 실험 및 고찰

제시한 알고리즘의 효율성을 알아보기 위해 실제로 반도체 생산으로 유명한 I 사에서 개발 중인 각종 회로에 대해 실험을 수행하였다. 본 실험실과 공동 연구하고 있는 I 사에 대해 그리고 실험에 사용한 회로의 정확한 명칭을 회사의 기밀상 밝힐 수 없음을 독자들의 양해를 구한다. 실험에 사용된 회로는 모두 표준 셀 레이아웃을 위한 것이며, 공동 연구중인 I 사가 보유한 최근의 배치기를 이용하여 최종 배치를 얻은 다음 제시한 알고리즘으로 그 배치를 개선하였다. 측정된 HP 모델을 근거로 한 총 배선길이이다. 표 1 에서 실험에 사용된 회로의 사양에 대해 보여준다.

표 1. 회로 사양

	#cells	#pads	#nets	#rows
Ckt1	1242	1451	1591	24
Ckt2	1500	556	1696	21
Ckt3	3027	1208	3135	37
Ckt4	3431	3391	4034	37
Ckt5	5127	4135	5727	49
Ckt6	6039	1003	6946	44

표 2 에서 제안한 알고리즘을 적용하기 전 총 배선 길이와 적용 후 얼마나 줄었는지, 즉 제안한 알고리즘의 효율을 보여준다. 참고로 알고리즘 적용 전의 총 길이는 I 사에서 구한 가장 좋은 결과임을 밝힌다.

표 2. 최적 인터리빙 적용 결과

	Opt_Int 적용전	Opt_Int 적용후	개선된 비율(%)
Ckt1	9.52051 e+06	9.26657 e+06	1.9
Ckt2	1.08782 e+07	1.06687 e+07	3.0
Ckt3	4.63846 e+07	4.35320 e+07	6.1
Ckt4	4.95299 e+07	4.78388 e+07	3.4
Ckt5	5.20145 e+07	5.04311 e+07	2.7
Ckt6	7.66929 e+07	7.39887 e+07	3.5

4. 결론

본 논문에서는 동적 프로그래밍 기법을 이용하여 미세 배치를 개선 할 수 있는 알고리즘을 제안하였다. 제안한 알고리즘의 핵심은 최적 인터리빙이라 불리는 기법의 사용인데 이는 각 행에 위치한 셀의 위치를 그 행 내에서 순서를 재 조정할 수 있도록 함으로써 총 배선길이를 줄인다. 제안한 알고리즘에 의하면 새로운 순서는 반드시 개선되기 때문에 이 알고리즘의 적용으로 인하여 결코 배치가 더 나빠지지는 않는다. 표 2 에 나타난 결과를 얻기 위해 평균적으로 사용된 CPU 시간은 Pentium IV 2GHz 기계 상에서 회로에 따라 7 초에서 99 초 소요되었다. 이는 제안한 알고리즘의 수행 속도가 충분히 빠름을 실험적으로 보여준다. 또한 제안한 알고리즘으로 인하여 전체적인 배선 길이의 길이가 평균 3.4% 향상되었다. 이는 각 행내에서의 위치변동만으로 얻은 결과임을 생각할 때, 즉 셀의 y 좌표값은 바뀌지 않은 점을 감안할 때 제안한 알고리즘으로 인한 각 넷의 HP 에서 x 성분의 감소는 이보다 훨씬 크다는 것을 알 수 있다. 만약 각 회로의 코어 영역의 모양이 정사각형이라면 각 넷의 x 성분의 감소는 약 7%라고 볼 수 있다.

최적 인터리빙의 개념을 한 행 내에서만 적용할 것이 아니라 임의의 수직 라인을 따라 적용한다면 셀들이 행의 위치를 바꿀 수 있다. 그러나 이렇게 적용할 경우 각 셀의 넓이가 다른 점 때문에 다소 셀이 행을 옮긴 후 셀 이동에 관계된 행들내에서 셀들의 위치가 또한 영향을 많이 받기 때문에 좀 더 복잡한 고려가 요구된다. 그러나 simulated annealing 기법에서 사용하고 있는 것 처럼 셀 크기가 다른 경우에 대한 가중치를 고려한다면 최적 인터리빙을 수직적으로도 적용할 수 있을 것으로 보인다. 제시한 최적 인터리빙의 기술을 확대 적용해서 수직라인에서의 최적화는 향후 연구과제로 남겨 둔다.

본 논문에서 얻은 실험결과는 한 행에서만 적용한 결과이므로 향후 연구 과제로 남긴 수직라인에서의 응용은 더 좋은 결과를 가져 올 수 있을 것이라 기대된다.

제안한 알고리즘을 좀 더 개선한다면 타이밍을 고려한 배치에서 critical 넷에 대한 HP 의 한계값이 주어질 경우 타이밍을 고려한 최적 인터리빙도 가능할 것으로 보인다.

참고문헌

[1] M. Wang and M. Sarrafzadeh, "Behavior of Congestion Minimization During Placement," Proc. of International Symposium on Physical Design, pp. 145-150, 1999.
 [2] M. Breuer, "Min-cut Placement," Design Automation and Fault-Tolerant Computing, pp. 343-382, 1977.
 [3] U. Lauther, "A Min-cut Placement Algorithm for General Cell Assemblies Based on a Graph Representation," J. of Digital Systems, Vol. IV, Issue 1, pp.21-34, 1980.
 [4] M. Terai, K. Takahashi and K. Sato, "A New Min-Cut Placement Algorithm for Timing Assurance Layout Design Meeting Net Length Constraint," Proc. of DAC,

1990.
 [5] A. E. Caldwell, A. B. Kahng, and Igor L. Markov, "Can Recursive Bisection Alone Produce Routable Placements?," Proc. of DAC, pp. 477-482, 2000.
 [6] N. R. Quinn and M. A. Breuer, "A Force Directed Component Placement Procedure for Printed Circuit Boards," IEEE Trans. on Circuits and Systems, Vol. CAS-26, pp. 377-388, 1979.
 [7] J. M. Kleinhans and G. Sigl and F. M. Johannes, "GORDIAN: A New Global Optimization / Rectangle Dissection Method for Cell Placement," Proc. of International Conference on CAD, pp.506-509, 1988.
 [8] Jens Vygen, "Algorithms for Large-Scale Flat Placement," Proc. of DAC, pp. 746-751, 1997.
 [9] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning," Proc. of DAC, pp. 269-274, 1998.
 [10] W. Swartz and C. Sechen, "New Alorithms for the Placement and Routing of Macro Cells," In Proc. of DAC, pp. 336-339, 1988.
 [11] Wern-Jieh Sun and Carl Sechen, "Efficient and Effective Placement for Very Large Circuits," IEEE Trans. on CAD, pp.349-359, 1995.
 [12] M. Sarrafzadeh and M. Wang, "NRG: Global and Detailed Placement," Proc. of ICCAD, pp. 532-537, 1997.
 [13] X. Yang, M. Wang, K. Egur, and M. Sarrafzadeh, "A Snap-on Placement Tool," Proc. of Intl. Symposium on Physical Design, pp. 153-158, 2000.
 [14] Sung-Woo Hur and John Lillis, "Mongrel: Hybrid Techniques for Standard Cell Placement", Proc. of ICCAD, pp. 165-170. 2000.